**19CSE100 Problem Solving and Algorithmic Thinking**

**Recursion**

There is an old nerd joke about recursion which goes as follows: *“To understand recursion you must first understand recursion”*

1. Write a *recursive* flowgorithmfunction add(a,b) to find the sum of two numbers given as input to the function.

Hint: Adding a and b is equivalent to adding 1, b times and when b becomes zero add a by returning a*.*

add(a,b)

if (b==0)

return a

else

sum = add(a,**b-1**)+1

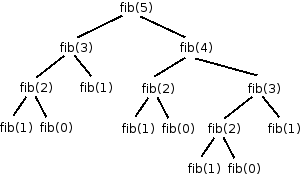
return sum

1. Write *recursive* flowgorithm function isPrime(n,divisor) that tests the primality of a given number.

Hint: The upper limit for the divisor shall be n/2 as you know. The recursive call keeps reducing the divisor checking the divisibility at each recursive call. If the divisibility check succeeds, n is not a prime number. If the divisor reaches 1, then n is a prime number. You can consider using a flag to capture the binary possibility.

1. Write a *recursive* flowgorithm function nthFibNum(n) that computes the nth Fibonacci number.

Hint: Each Fibonacci number follows a pattern that is fib(n) = fib(n-1)+fib(n-2) with seed values as fib(0)=1 and fib(1)=1 where fib(n) represents nth Fibonacci number. Using the above pattern if you notice taking fib(5) as an example as shown below, the seed values becomes base cases.



1. Write a *recursive* flowgorithm function isEven(num) to find whether a given number is even.

Hint: If you successively subtract 2 from num and end up with a zero, num is an even number. If the successive subtraction ends with a 1, num is an odd number. Isn’t it?! So your recursive call should be isEven(num-2) till either 1 or 0 reaches which happens to be the base cases to decide the outcome.

1. Write a *recursive* flowgorithm function printEvenOdd(start, end) that prints even or odd numbers in a given range. Depening on your start value either even or odd gets printed. If you give start = 1 then 1,3,5,7,9,... should get printed till the end.

Hint: Each successive recursive call should increment the start by 2 after printing it. The recursion should proceed as long as start is less than or equal to end.